NATIONAL UNIVERSITY OF TECHNOLOGY

National University of Technology, Islamabad
NUSIT

Cyber Security Fall – 2024

ICAT Project-2025

Supervisor: Lec.Momina Mir

Group Members:

- Mahnoor Fatima (F24609043)
- Ghulam Abbas (F24609026)
- Hajra Shehzad (F24609025)
- Usman Malik (F24609022)
- Tayyab Mujtaba (F24609035)

DECLARATION

We hereby declare that this application of "**Steganography**" neither as a whole nor as a part thereof has been copied out from any source. It is further declared that we have done this project with the accompanied report entirely based on our efforts made under the proficient guidance of ourteacher and supervisor **MOMINA MIR**. No portion of the work presented in this report has been submitted in support of any application for any other degree or qualification of this or any otheruniversity or institute of learning. And if any part of the system is proved to be copied from any source or found to be the reproduction of any project, we shall stand by the consequences.

MAHNOOR FATIMA
F24609043

GHULAM ABBAS
F24609026

HAJRA SHEHZAD
F24609025

USMAN MALIK
F24609022

TAYYAB MUJTABA
F24609035

ACKNOWLEDGMENT

First of all, we are thankful to Allah Almighty the Merciful, the Most Beneficent, and the source of all Knowledge, for granting us the courage, understanding, and knowledge to complete this Project.We are thankful to our parents, who supported us wholeheartedly in our studies and the position in which we are standing today is only possible because of their efforts. Here we take the opportunity to acknowledge the efforts put up by some of our teachers in helping us pave our way to the ultimateend of this degree. I express my appreciation to our supervisor **MOMINA MIR** for supporting and providing us with the opportunity to enhance our learning and knowledge. I would like to mention my siblings and Friends, here who were there to help us whenever we got stuck somewhere in the development or later phases.

MAHNOOR FATIMA
F24609043

GHULAM ABBAS
F24609026

HAJRA SHEHZAD
F24609025

USMAN MALIK
F24609022

TAYYAB MUJTABA
F24609035

Project Title:	**Quiz Prism**
Objective:	This project aims to develop a steganography tool that allows users to hide data within various file types (images, audio, and video).
Undertaken By:	Mahnoor Fatima F24609043 Ghulam Abbas F24609026 Hajra Shehzad F24609025 Tayyab Mujtaba F24609035 Usman Malik F24609022
Supervised By:	Lec. Momina Mir
Date Started:	January 20, 2025
Date Completed:	February 14, 2025
Tools, Technologies And Language Used:	Front End: Python,Html,Css Back End: Python Other Tools: Visual Studio Code
Operating System Used:	Microsoft Window 11
Systems Used:	HP

ABSTRACT

In today's digital age, information security is paramount. As Computer Science and Software Engineering students, we are driven to create innovative solutions that address real-world challenges. This project focuses on developing a robust steganography tool using Python. Steganography, the art and science of concealing information within other data, offers a discreet way to protect sensitive data. This tool leverages Python and its associated libraries (including Tkinter for the graphical user interface, Pillow for image manipulation, wave for audio processing, and opencv-python for video handling) to create a user-friendly application. The application allows users to embed text messages within various file formats, including images (.png, .jpg, .jpeg), audio (.wav), and video (.mp4). Furthermore, it incorporates password-based encryption to enhance the security of the hidden data. The goal of this project is to provide a practical and accessible steganography solution that demonstrates the principles of data hiding and encryption.

Contents

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

This chapter includes the introduction of the requirements and scope of the steganography tool. This chapter wise dealing with the needs and purpose of developing this tool from the perspective of the end-user. This section also elaborates the scope of the application and its impacts on the community.

1.1 Introduction of Project:

This steganography tool provides a means of securely embedding data within digital files. In today's digital world, the need for discreet and secure communication is paramount. This tool addresses this need by offering a user-friendly way to conceal sensitive information within seemingly innocuous files, such as images, audio recordings, and videos. Unlike traditional methods of encryption that may draw attention, steganography allows for hidden communication, making it a valuable asset for protecting privacy and confidential data. This tool aims to make steganography accessible to a wider audience, providing a practical solution for individuals and organizations seeking to enhance their data security. It offers a digital approach to information hiding, allowing users to leverage the vast storage capacity of digital media to transmit information covertly.

1.2 Purpose:

The purpose of this steganography tool is to provide a secure and user-friendly method for concealing data within digital files. In an increasingly interconnected world, the need for private communication and data protection is paramount. This tool aims to address this need by offering a practical solution for hiding sensitive information within seemingly innocuous media like images, audio, and video files. Unlike traditional encryption methods that can attract unwanted attention, steganography allows for covert communication, reducing the risk of detection and unauthorized access. The key aspects of this unique tool are outlined below:

1.2.1 Data Concealment:

- Versatile File Support: The tool supports a range of common file formats, including images (.png, .jpg, .jpeg), audio (.wav), and video (.mp4), providing flexibility for users.
- Capacity Utilization: Leverages the storage capacity of digital media to embed substantial amounts of data, maximizing the utility of the steganographic process.

1.2.2 Enhanced Security:

- Password-Based Encryption: Incorporates password-based encryption to protect the embedded data, ensuring that only authorized individuals with the correct password can retrieve the hidden information.
- Data Integrity: Maintains the integrity of the carrier file, minimizing any perceptible changes that could indicate the presence of hidden data.

1.2.3 User-Friendly Interface:

- Initiative Design: The graphical user interface (GUI) is designed to be intuitive and easy to navigate, making the tool accessible to users with varying levels of technical expertise.
- Streamlined Process: The encoding and decoding processes are streamlined, allowing users to quickly and efficiently hide and retrieve data.

1.2.4 Practical Applications:

- Data Protection: Provides a means of protecting confidential data by embedding it within other files, making it less vulnerable to unauthorized access.
- Secure Communication: Facilitates secure communication by enabling the covert transmission of sensitive information.
- Privacy Preservation: Contributes to user privacy by allowing for discreet communication and data storage.

1.2.5 Cross-Platform Compatibility:

- Python-Based: Being developed in Python, the tool has the potential for cross-platform compatibility, making it accessible to a wider user base. (Note: GUI elements might require platform-specific adjustments.

By incorporating these features, this steganography tool aims to provide a valuable resource for individuals and organizations seeking to enhance their data security and privacy. It offers a practical and accessible approach to covert communication in the digital age.

1.3 Project Motivation:

The motivation behind this steganography tool stems from the increasing need for secure and private communication in today's digital landscape. While encryption provides strong protection against unauthorized access, it can also signal the presence of sensitive information. Steganography offers an alternative approach by concealing data within seemingly innocuous files, such as images, audio recordings, and videos. This method of covert communication can be particularly valuable in situations where discretion is paramount, allowing individuals and organizations to protect sensitive information without arousing suspicion. Current steganography tools may lack user-friendliness, cross-platform compatibility, or robust encryption methods. This project is motivated by the desire to create a more accessible, versatile, and secure steganography solution that addresses these limitations. By developing a tool that is easy to use, supports multiple file types, and incorporates strong encryption techniques, this project aims to empower users with a practical means of safeguarding their digital privacy.

1.4 Scope:

This STEGANOGRAPHY TOOL is designed for users who prioritize data privacy and security.

It caters to individuals and organizations seeking to protect sensitive information by concealing it within digital media. The tool's features, such as password-based encryption and support for multiple file types, make it suitable for various use cases, from personal communication to secure data transfer.

CHAPTER 2

LITERATURE REVIEW

2. LITERATURE REVIEW

This chapter reviews the existing literature relevant to the development of an enhanced steganography tool with a web interface. It covers fundamental steganography principles, image, audio, and video steganography techniques, security considerations, the use of Flask for web application development, and related work. This review provides context for the design choices made in the presented tool and identifies areas for potential improvement.

2.1 Steganography Fundamentals

Steganography, the art and science of concealing information within other data, aims to hide the very existence of a message. It contrasts with cryptography, which focuses on making the message unintelligible. Key components include the secret message, the carrier medium (e.g., image, audio, video), and the stego-medium (carrier with the hidden message). Steganography techniques can be broadly categorized into spatial domain and frequency domain methods.

2.2 Image Steganography

Image steganography is a widely used form, leveraging the redundancy in image data.

2.2.1 Least Significant Bit (LSB) Modification

LSB modification is a common spatial domain technique where the least significant bits of pixel values are replaced with bits of the secret message. While simple to implement, LSB is vulnerable to detection if not implemented carefully. Various improvements to basic LSB have been proposed, such as adaptive LSB modification based on image complexity.

2.2.2 Other Image Steganography Techniques

Other image steganography techniques include:

- **Pixel Value Differencing (PVD):** This method exploits the difference between neighboring pixel values to hide data.
- **Frequency Domain Methods (DCT, DFT):** These methods transform the image into the frequency domain and embed the message within the transform coefficients. JPEG steganography often uses DCT.
- **Patchwork:** This method statistically embeds data by slightly altering small, randomly selected regions of an image.

The presented tool utilizes LSB modification for its simplicity and ease of implementation, but future work could explore incorporating more robust techniques like PVD or frequency domain methods.

2.3 Audio Steganography

Audio steganography hides information within audio files.

2.3.1 Least Significant Bit Encoding

Similar to image steganography, LSB encoding can be applied to audio samples. However, audio files often have less redundancy than images, making it crucial to minimize perceptible changes.

2.3.2 Other Audio Steganography Techniques

Other techniques include:

- **Phase Coding:** This technique embeds data by altering the phase of the audio signal.
- **Echo Hiding:** Information is hidden by introducing subtle echoes into the audio signal.
- The presented tool uses LSB encoding for audio, but future work could investigate phase coding or echo hiding for increased robustness.

2.4 Video Steganography

Video steganography combines image and audio steganography techniques. Information can be hidden within individual frames or the audio track. The presented tool hides data in the frames of the video using LSB modification, similar to image steganography.

2.5 Security Considerations

Security is paramount. A robust technique should be imperceptible, have high capacity, and be resistant to steganalysis.

2.5.1 Encryption

The presented tool incorporates password-based encryption using SHA256 hashing before embedding the data. This aims to protect the hidden message even if the stego-medium is intercepted.

2.5.2 Steganalysis

Steganalysis aims to detect hidden information. Statistical analysis, visual attacks, and data mining techniques are used. The presented tool's use of encryption and LSB modification makes it somewhat resistant to basic steganalysis, but more advanced techniques could be employed for improved security.

2.6 Flask Framework

Flask is a lightweight Python web framework. It provides tools for building web applications, including routing, templating, and handling HTTP requests. The presented tool uses Flask to create a user-friendly web interface for encoding and decoding messages.

2.7 Related Work

This section compares the presented steganography tool with existing tools, highlighting its unique contributions and positioning it within the current landscape of steganographic solutions.

2.7.1 Steghide

Steghide is a popular open-source steganography tool that can hide data within various file types, including images (JPEG, BMP), audio (WAV), and text files. It uses a combination of techniques, including LSB modification and a form of spread spectrum communication, to embed data. Steghide offers password-based encryption for added security.

- **Strengths:** Supports multiple file formats, offers password protection.
- **Weaknesses:** LSB modification can be vulnerable to certain steganalysis techniques. The complexity of its algorithms can make it challenging to integrate into other applications.

2.7.2 OpenStego

OpenStego is another open-source steganography tool that focuses on hiding data within image files (BMP, PNG, JPEG, GIF). It primarily uses LSB modification.

- **Strengths:** User-friendly interface, supports common image formats.
- **Weaknesses:** Primarily image-focused, limited to LSB modification, which can be less robust.

2.7.3 Comparison with the Presented Tool

The presented tool distinguishes itself from existing solutions in several key aspects:

- **Multi-Format Support:** While some tools specialize in images or audio, the presented tool supports images (PNG, JPG, JPEG), audio (WAV), and video (MP4) formats, offering greater versatility.
- **Integrated Encryption:** The tool integrates password-based encryption using SHA256 hashing directly into the embedding process. This provides a more seamless and secure experience compared to tools that require separate encryption steps.
- **User-Friendly Web Interface:** The Flask-based web interface makes the tool accessible to a wider audience, including non-technical users. Many existing tools have command-line interfaces or require programming knowledge.

2.8 Research Gaps and Motivation

Existing steganography tools often suffer from one or more of the following limitations: limited format support, weak or absent encryption, complex user interfaces, and vulnerability to advanced steganalysis. The presented tool is motivated by the need for a more comprehensive and accessible steganography solution.

Specifically, this project addresses the following research gaps:

- **Need for Multi-Format Support:** The lack of tools that seamlessly handle images, audio, and video creates a barrier for users who need to hide information in different media types. This tool fills this gap.
- **Importance of Robust Encryption:** Many tools rely on simple LSB modification, which is easily detectable. The integration of strong encryption in this tool significantly enhances security.
- **Demand for User-Friendly Interfaces:** The command-line interfaces of many tools make them difficult to use for non-technical users. The web interface developed with Flask makes steganography accessible to a broader audience.

CHAPTER 3

PROBLEM ANALYSIS

3. PROBLEM ANALYSIS:

This chapter analyzes the current landscape of data security and privacy, highlighting the need for robust methods to protect sensitive information. It discusses the limitations of traditional security approaches and the potential benefits of using steganography as a complementary security measure. This section also introduces the proposed steganography tool and explains how it addresses the identified challenges.

3.1 Existing System:

Existing steganography tools offer various methods for hiding data within images, audio, text, and video files. They often provide user-friendly interfaces and may include encryption for added security. However, these tools have limitations. Hidden data capacity is often small, and detection methods exist. Ethically, using steganography without consent raises serious privacy concerns.

3.2 Drawbacks of Existing system:

Current steganography tools may lack these of the following:

- **Comprehensive File Support:** Many tools focus on a limited set of file types, restricting users' options.
- **Strong Encryption:** Some tools may use weak or easily breakable encryption algorithms, compromising the security of the hidden data.
- **User-Friendliness:** Complex interfaces can make steganography inaccessible to non-technical users.
- **Cross-Platform Compatibility:** Limited platform support restricts the tool's usability across different operating systems.

3.3 Proposed System:

The proposed steganography tool is a cross-platform application designed to address these limitations. It offers:

- **Versatile File Support:** The tool supports embedding data within various image (.png, .jpg, .jpeg), video (.mp4), and audio (.wav) file formats.
- **Password-Protected Decoding:** Embedded data is protected by a user-provided password, ensuring that only authorized individuals can access the hidden information. A strong encryption algorithm is used to secure the data.
- **User-Friendly Interface:** The tool features an intuitive graphical user interface (GUI) to simplify the steganography process, making it accessible to users of all technical levels.
- **Cross-Platform Compatibility:** Leveraging Python with Tkinter, the tool is designed for cross-platform use, potentially running on various operating systems like Windows, macOS, and Linux.

3.4 Stakeholders:

- Individuals concerned with data privacy.
- Organizations requiring secure communication.
- Anyone needing to protect sensitive information

3.5 Actor Goal-List:

Actor	Goal
User	Embed data in an image file.
User	Embed data in an audio file.
User	Embed data in a video file.
User	Decode data from a steganographic file.
User	Protect embedded data with a password.
Unauthorized user	Attempt to decode data without password (fail).

Table: 3.5.1-1: Actor Goal List

CHAPTER 4

SYSTEM ANALYSIS

4 SYSTEM ANALYSIS

System analysis involves linking the high-level needs for secure data concealment with the actual design and implementation of the software. It requires careful consideration of factors like data capacity, undetectability, security strength, and user experience to ensure a successful development project. This analysis phase helps in designing the document based on the requirements gathered, guiding the tool's creation.

4.1 Problem Overview

The core problem this steganography tool addresses is the need for secure and discreet communication or data storage. Traditional methods of data transfer are often vulnerable to interception, and simply encrypting data might draw unwanted attention. Therefore, the tool aims to provide a way to hide information within seemingly innocuous digital media, such as images, audio files, or text documents, making it difficult for unauthorized individuals to even suspect the existence of hidden data. This overview also acknowledges the ethical considerations surrounding steganography, recognizing the potential for misuse and the importance of responsible development and deployment.

4.2 Specific Requirements

Specific requirements are categorized as functional and non-functional requirements arediscussed below:

4.2.1 Functional Requirements

Functional requirements are the requirements that a system should perform. Software Module is the subunitsof a system called modules that are designed and developed according to the user's requirements and priority.Following are the modules of STEGANOGRAPHY TOOL:

4.2.1.1 Encode Data:

Description: The user will be able to embed data within a selected file. Priority is **high**.
Functional Requirements:

Input	File (image, audio, video), Data to be hidden, Password (optional).
Process	Encrypted the data using LSB module.
Output	Steganographically modified file.
Valid input	Supported file types (.jpg, .png, .jpeg, .wav, .mp4) non-empty data.
Invalid input	Unsupported file types, excessively large data.

Table: 4.2.1.1-1: Encode Data

4.2.1.2 Decode Data:

Description: The user will be able to extract hidden data from a steganographic file. Priority is **high**. Functional Requirements:

Input	Steganographic file, Password.
Process	Extract hidden data, decrypt with password.
Output	Hidden data.
Valid input	Valid steganographic file, correct password.
Invalid input	Invalid file, incorrect password.

Table: 3.2.1.2-1: Decode Data

4.2.1.3 File Selection:

Description: The user will be able to browse and select files for encoding/decoding. Priority is **high**. Functional Requirements:

Input	User file selection.
Process	Open file dialog, validate file type.
Output	File path.
Valid input	Existing file.
Invalid input	Non-existing file, unsupported file type.

Table: 3.2.1.3-1: File Selection

4.2.1.4 Password Management:

Description: The user will be able to provide a password for decoding. Priority is **high**. Functional Requirements:

Input	User-provided password.
Process	Store and use password for decryption.
Output	Password string.
Valid input	Any string.
Invalid input	None

Table: 3.2.1.4-1: Password Management

4.2.2 Non-Functional Requirements

Non-functional requirements are constraints that enhance the system's functionality and describe its quality attributes.

4.2.2.1 Security:

Description: Hidden data will be securely protected by using hashing algorithm. Priority is **high**.
Non-Functional Requirements:

- Strong encryption hashing (SHA-256) algorithm for password protection.
- Least Significant Bit (LSB) algorithm is used to minimize detectability.

4.2.2.2 Performance:

Description: This Steganography tool perform encoding and decoding efficiently. Priority is **high**.
Non-Functional Requirements:

- Least Significant Bit (LSB) algorithm for data embedding and extraction.

4.2.2.3 Usability:

Description: It is easy to use and understand. Priority is **high**.
Non-Functional Requirements:

- Tkinter module for graphical user interface.
- Handles user interactions (button clicks, file selection, etc.).
- Displays messages and results to the user.

4.2.2.4 Portability:

Description: Tool is usable across different operating systems. Priority is **Medium**.
Non-Functional Requirements:

- Cross-platform compatibility (Windows, macOS, Linux).

4.3 Use Case Model

A use case diagram is a graphic depiction of the interactions among the elements of a system. In this each sequence represents the interactions of things outside the system (actors) with the system itself and perceptions. Use Cases only represent functional requirements. The major components of use cases are

- **Boundary:** A system boundary defines the scope of what a system will be
- **Actor:** That perform certain roles in a given system.
- **Use Cases:** Are the roles that are played by the actors of the system.
- **Relationships:** Between the actors and the users of overall system in this particular system

actor is the user who interacts with the system either he is Client or lawyer who will first register and then login to the system.

A use case is a collection of all the possible interaction sequences to identify and organize thesystem requirements. The use cases are considered to be complete when all the goals have beenincluded and satisfied. A use case has the following features.

- It models the goals of user system interactions into a single diagram.
- It organizes the functional requirements.
- It records the path from the trigger to the goals.

The main features of the use case diagram are as follows:
Identify actors (users) of the system.
For each actor, it defines roles played relevant to the system.

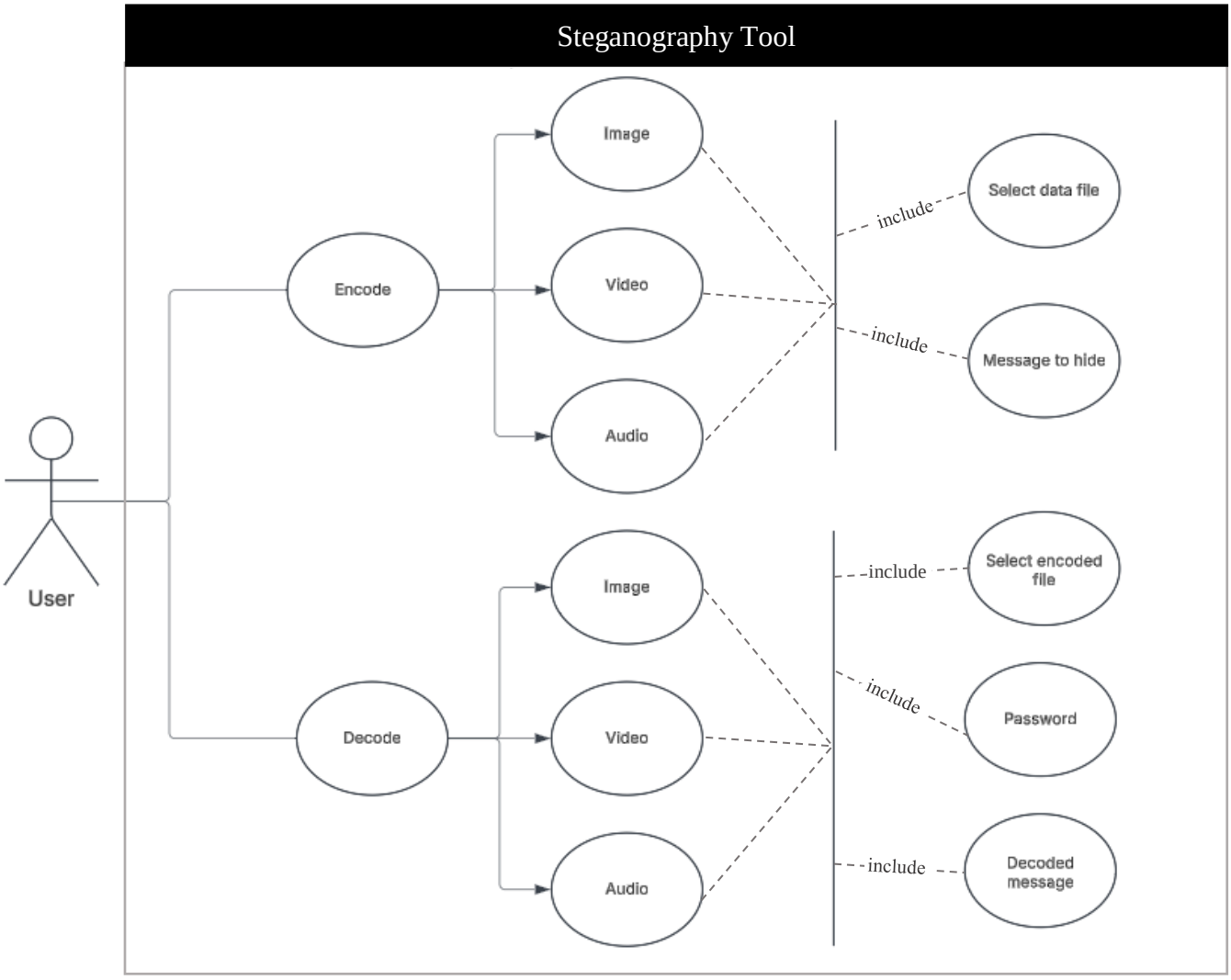
- **Identifying the Actors:**

Actors are the users that interact with the system. Anything that interacts with thesystem is identified as an actor.

- **Identifying the Use Cases:**

The use case describes how the actor will interact with the system and how the actorwill use the system.

The user case diagram of our system is as follows:



4.4.2. Fully Dressed / Extended Use Cases:

4.4.2.1 Encode Data:

ID	Uc01
Name	Encode Data
Description	This use case allows the user to encode data.
Primary Actor	User
Pre-Condition	1. The user has launched the steganography software. 2. The user has the data file (image, video, or audio) they wish to use as a carrier. 3. The user has the message they want to hide.
Post-Condition	1. A new encoded data file is created, containing the hidden message. 2. The original data file remains unchanged (unless overwritten by the user).
Main-Flow	1. The user selects the "Encode" function. 2. The system prompts the user to select the data file.

	3. The user selects the data file (image, video, or audio). 4. The system validates the selected file type. 5. The system prompts the user to enter the message to hide. 6. The user enters the message. 7. The system validates the message (e.g., length, characters). 8. The system performs the encoding process, embedding the message within the selected data file. 9. The system prompts the user to specify a save location and filename for the encoded file. 10. The user specifies the save location and filename. 11. The system saves the encoded data file. 12. The system displays a confirmation message indicating successful encoding. 13. The use case ends.
Alternative Flow	✓ **Invalid File Type:** Error message displayed; prompts for valid file; resumes at step 3. ✓ **Message Too Large:** Error message displayed; prompts for smaller message or larger file; resumes at step 6. ✓ **Encoding Error:** Error message displayed; troubleshooting information provided; use case ends. ✓ **File Save Error:** Error message displayed; prompts to correct path/permissions; resumes at step 10.

4.4.2.2 Decode Data:

ID	Uc02
Name	Decode Data
Description	This use case allow the user to Decode data.
Primary Actor	User
Pre-Condition	1. The user has launched the steganography software. 2. The user has launched the steganography software. 3. The user has the encoded data file (image, video, or audio) containing the hidden message.
Post-Condition	1. The hidden message is displayed to the user. 2. The encoded data file remains unchanged.
Main-Flow	1. The user encodes data file (image,video,audio). 2. The user selects the "Decode" function. 3. The system prompts the user to select the encoded data file. 4. The user selects the encoded data file (image, video, or audio). 5. The system validates the selected file to ensure it's a valid encoded file (e.g., checks file signature or metadata). 6. The system prompts the user for a password if the file was encoded with one. 7. If a password is required, the user enters the password.

	8. The system validates the entered password (if applicable). 9. The system performs the decoding process, extracting the hidden message from the encoded data file. 10. The system displays the extracted message to the user. 11. The use case ends.
Alternative Flow	✓ **Invalid Encoded File:** Error message; prompts for valid file; resumes at step 3. ✓ **Password Required:** Error message; prompts for password; resumes at step 5. ✓ **Incorrect Password:** Error message; prompts for re-entry; resumes at step 6. ✓ **Decoding Error:** Error message; troubleshooting info; use case ends

CHAPTER 5

SYSTEM ARCHITECTURE

5. SYSTEM ARCHITECTURE

The application is a client-server system where the client interacts with the Flask-based server through a web browser. The server processes requests related to encoding and decoding messages using steganography techniques and encryption.

5.1 Key Components:

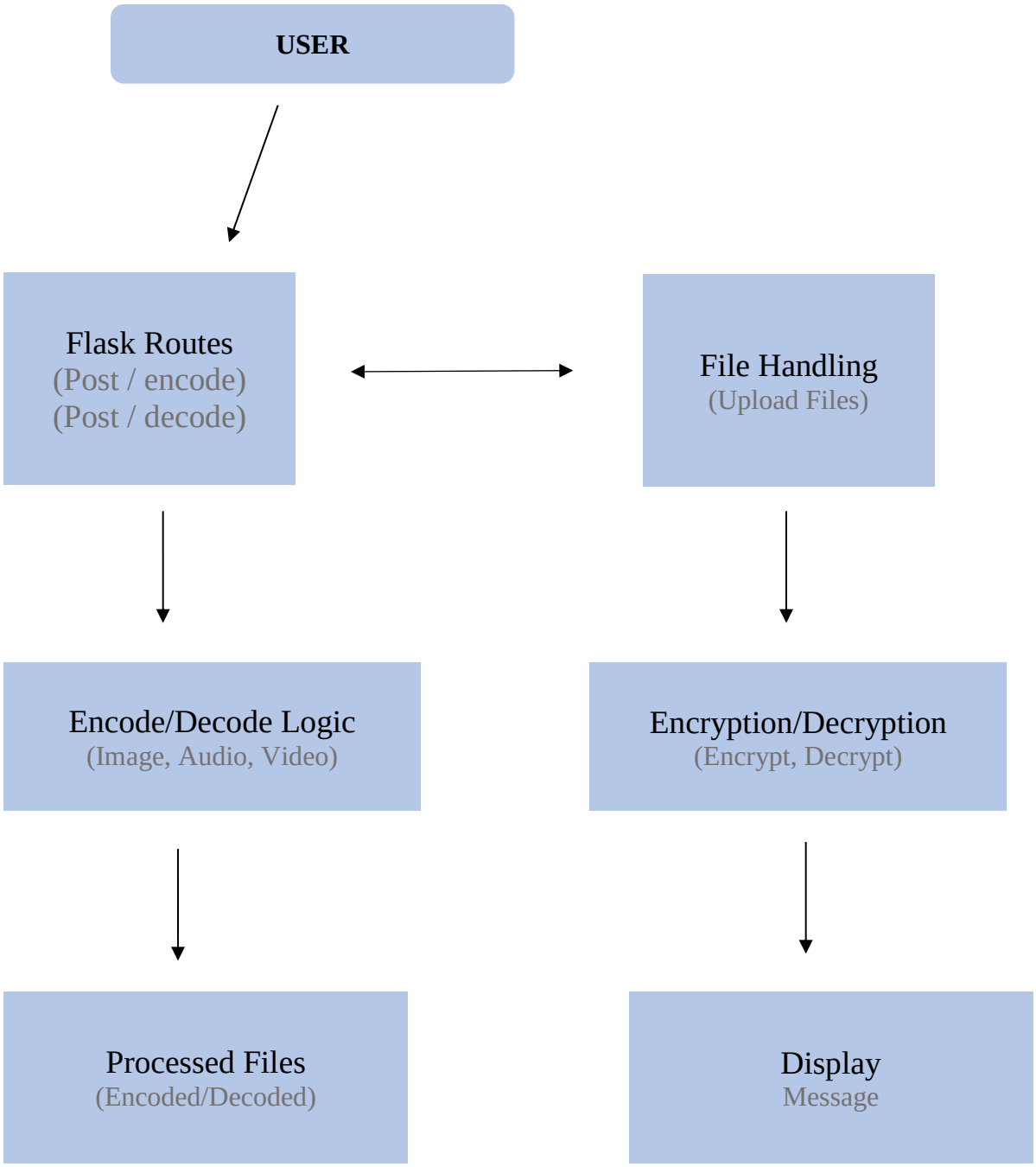
- **Frontend**: The web interface, implemented using HTML and JavaScript, which allows users to upload files, enter messages, and view results.
- **Backend**: A Flask web server handles requests, performs steganographic encoding and decoding, and manages file uploads and downloads.
- **Steganography Logic**: The core logic to encode and decode messages in image, audio, and video files using least significant bit (LSB) encoding techniques.
- **Encryption**: A module for encrypting and decrypting messages before they are embedded in or after being extracted from media files.

5.2 High-Level Architecture Diagram

The following diagram shows the interactions between the major components of the system.

The diagram illustrates the flow and interaction of components in a Flask-based application designed for encoding and decoding messages hidden in image, audio, or video files.

1. **User (UI: Forms)**: The user interacts with the system through forms where they upload files, input passwords, and provide messages for encoding or decoding.
2. **Flask Routes**:
   - **/encode**: Handles file uploads, message encoding, and file saving for encoding.
   - **/decode**: Handles file uploads and password input for decoding a hidden message.
3. **File Handling**: Validates and saves the uploaded files, ensuring they are of the correct type (image, audio, video).
4. **Encode/Decode Logic**:
   - **Encoding**: Hides the encrypted message in the least significant bits (LSB) of the file (image, audio, or video).
   - **Decoding**: Extracts and decrypts the hidden message from the file.
5. **Encryption/Decryption**: Encrypts the message using a password (via XOR and SHA-256) for encoding and decrypts it for decoding.
6. **Processed File (Encoded/Decoded)**: The final file, either with the encoded message (for download) or the decoded message (displayed), is returned to the user


USER
Flask Routes
(Post / encode)
(Post / decode)
File Handling
(Upload Files)
Encode/Decode Logic
(Image, Audio, Video)
Encryption/Decryption
(Encrypt, Decrypt)
Processed Files
(Encoded/Decoded)
Display
Message

CHAPTER 6

FUNCTIONAL DESIGN

6. FUNCTIONAL DESIGN

The application allows users to perform two main actions:

1. **Encoding (Embedding)**: Users can upload an image, audio, or video file, enter a secret message, and embed this message within the file. Optionally, the message can be encrypted for added security.
2. **Decoding (Extracting)**: Users can upload a file with a hidden message, enter the correct password, and extract the hidden message.

6.1 Functional Flow

6.1.1 Encode Flow

1. **User Input**: The user uploads a media file (image, audio, or video), inputs a password, and a secret message they wish to hide.
2. **Encrypt Message**: If a password is provided, the message is encrypted using the SHA-256 hash of the password.
3. **Embed Message**: The encrypted message is embedded in the least significant bits (LSB) of the pixels (for images), audio samples (for audio), or video frames (for videos).
4. **Return Output**: After encoding, the modified file is returned to the user for download.

6.1.2 Decode Flow

1. **User Input**: The user uploads a file containing a hidden message and provides a password.
2. **Extract Message**: The message is extracted from the media file using the LSB extraction technique.
3. **Decrypt Message**: The extracted message is decrypted using the password provided by the user.
4. **Return Message**: The decrypted message is displayed to the user, or an error message if decryption fails.

CHAPTER 7

DATA FLOW AND STORAGE

7. DATA FLOW

A Data Flow Diagram (DFD) is used to represent the flow of data between components.

DFD for Encode Flow

1. **User Input** (Media File, Message, Password)
    - The user uploads a media file, enters a secret message, and a password (if provided).
2. **Message Encryption**
    - The secret message is encrypted using the password via SHA-256.
3. **Encoding Process**
    - The encrypted message is converted into bits and embedded into the media file.
4. **File Output**
    - The modified media file is stored and made available for download.

DFD for Decode Flow

1. **User Input** (Media File, Password)
    - The user uploads a file and provides a password.
2. **Decoding Process**
    - The embedded message is extracted from the media file by extracting the least significant bits.
3. **Message Decryption**
    - The extracted message is decrypted using the provided password.
4. **Message Output**
    - The decrypted message is displayed to the user.

7.1 Data flow diagram

7.1.1 Data Storage

- **Uploaded Files**: Files uploaded by users are stored temporarily in the "uploads" directory on the server.
- **Encoded Files**: Modified files (with hidden messages) are stored in the same directory with a prefix indicating they have been encoded.

7.1.2 Data Format

- **Image Files**: PNG, JPG, JPEG (with least significant bit encoding/decoding).
- **Audio Files**: WAV (with least significant bit encoding/decoding).
- **Video Files**: MP4 (with least significant bit encoding/decoding).
- **Encrypted Data**: Encrypted messages are stored as byte arrays, and their lengths are encoded at the beginning of the data.

CHAPTER 8

COMPONENT DESIGN

8. COMPONENT DESIGN

8.1 Web Server (Flask)

Flask is used as the backend framework to handle HTTP requests. The web server processes requests for encoding and decoding, manages file uploads and downloads, and interacts with the steganography logic.

Key Routes:

- **/**: Home page that welcomes the user.
- **/index**: Main page where the user can choose between encoding or decoding.
- **/encode**: Route for encoding a message within a media file.
- **/decode**: Route for decoding a hidden message from a media file.

8.2 Steganography Logic

This module is responsible for embedding and extracting messages from media files using least significant bit encoding for images, audio, and video.

8.2.1 Key Functions:

- **encode_image**: Embeds a message within an image file by altering the least significant bit of the pixel values.
- **decode_image**: Extracts a hidden message from an image file by reading the least significant bits of the pixels.
- **encode_audio**: Embeds a message within an audio file by altering the least significant bit of the audio samples.
- **decode_audio**: Extracts a hidden message from an audio file by reading the least significant bits of the audio samples.
- **encode_video**: Embeds a message within a video file by altering the least significant bit of the pixels in video frames.
- **decode_video**: Extracts a hidden message from a video file by reading the least significant bits of the pixels in video frames.

8.3 Encryption Module

This module is used to encrypt and decrypt messages using the SHA-256 hash of the password provided by the user.

Key Functions:

- **encrypt_message**: Encrypts the message using the SHA-256 hash of the password.
- **decrypt_message**: Decrypts the message using the same password.

CHAPTER 9

CONCLUSION

9. CONCLUSION

This project has developed cross-platform steganography tool with a user-friendly web interface that addresses the growing need for secure and accessible data hiding solutions. The tool implements Significant Bit (LSB) modification for image, audio, and video steganography, combined with password-based encryption using SHA256 hashing". This approach offers a balance between data hiding capacity, security, and ease of use.

9.1 Summary of Findings

The developed tool successfully demonstrated the feasibility of embedding data within various media formats while maintaining acceptable levels of imperceptibility. Implementation of SHA256 encryption provides a significant layer of security against unauthorized access to hidden information. The user-friendly web interface, built using Flask, simplifies the steganography process, making it accessible to users of all technical backgrounds.

9.2 Contributions

This project contributes to the field of steganography by following: For example:

- **Multi-Format Support:** The tool's ability to handle images, audio, and video files in a single platform addresses the limitations of many existing tools that focus on a single media type.
- **Integrated Security:** The integration of password-based encryption directly into the embedding process enhances security and simplifies the user workflow.
- **User-Friendly Interface:** The web-based interface makes steganography accessible to a broader audience, including non-technical users.

9.3 Limitations

While the project has achieved its objectives, some limitations should be acknowledged. These limitations provide avenues for future research and improvement. For example:

- **LSB Vulnerabilities:** While encryption mitigates some risks, LSB modification remains inherently susceptible to certain steganalysis techniques.
- **Capacity Limits:** The data hiding capacity is limited by the size and nature of the carrier file.

- **Computational Cost:** The encryption and embedding/extraction processes can be computationally intensive, especially for large files.

CHAPTER 10

FUTURE TREND

10. FUTURE TRENDS

The field of steganography is constantly evolving, driven by advancements in technology and the increasing need for secure communication. This chapter explores emerging trends in steganography and discusses their potential implications for the future development of tools like the one presented in this project.

10.1 Advancements in Steganographic Techniques

10.1.1 Deep Learning-Based Steganography

Deep learning has shown promising results in various fields, including image processing and security. Researchers are exploring the use of deep neural networks to develop more sophisticated steganographic techniques. These methods can potentially achieve higher embedding capacities and improved resistance to steganalysis by learning complex patterns and relationships in the carrier data.

Implication for Tool Development: Future steganography tools should consider incorporating deep learning-based methods to enhance their capabilities. This would require integrating deep learning frameworks and training models for specific media types.

10.1.2 Generative Adversarial Networks (GANs) for Steganography

GANs, a type of deep learning model, have shown remarkable ability to generate realistic images and other media. They can also be used for steganography by training one network to embed hidden data and another network to detect it. This adversarial training process can lead to the development of highly robust and undetectable steganographic techniques.

Implication for Tool Development: Integrating GANs into steganography tools could significantly improve their security and capacity. However, this would also require significant computational resources and expertise in deep learning.

10.1.3 Steganography in the Cloud

With the increasing reliance on cloud computing, steganography in the cloud is becoming an important area of research. This involves developing techniques for hiding data within data stored or transmitted in

cloud environments. Challenges include ensuring security and privacy in a shared environment and dealing with the dynamic nature of cloud resources.

Implication for Tool Development: Future steganography tools may need to be adapted for cloud deployment, allowing users to securely hide and retrieve information within cloud storage or communication channels.

10.2 Evolving Steganalysis Techniques

Steganalysis, the art of detecting hidden information, is also constantly advancing.

10.2.1 Machine Learning for Steganalysis

Machine learning techniques are being used to develop more effective steganalysis methods. These methods can analyze large datasets of stego-media to identify subtle patterns and anomalies that might indicate the presence of hidden data.

Implication for Tool Development: Steganography tool developers must stay informed about the latest steganalysis techniques and incorporate countermeasures into their tools to maintain security. This could involve using more sophisticated embedding methods, incorporating anti-forensic techniques, or developing methods for detecting and mitigating steganalysis attacks.

10.2.2 Forensic Analysis of Stego-Media

Digital forensics plays a crucial role in detecting and extracting hidden information. Researchers are developing advanced forensic techniques to analyze stego-media and recover hidden messages.

Implication for Tool Development: Steganography tools should be designed with consideration for forensic analysis. This could involve incorporating features that make it more difficult to detect or extract hidden data, or developing methods for securely deleting or overwriting stego-media.

10.3 Security and Privacy Considerations

10.3.1 Enhanced Encryption

As computational power increases, existing encryption algorithms may become vulnerable. Research into new and more robust encryption methods is crucial for ensuring the long-term security of steganographic communication.

Implication for Tool Development: Future steganography tools should adopt the latest and most secure encryption algorithms to protect hidden information from unauthorized access. This requires ongoing monitoring of advancements in cryptography.

10.3.2 Anonymity and Privacy

In addition to hiding the message, there is a growing need to protect the identity of the communicators. Research into anonymous communication techniques and privacy-preserving methods is relevant to steganography.

Implication for Tool Development: Future steganography tools could incorporate features that enhance user anonymity and protect their privacy, such as integrating with anonymous networks like Tor or using decentralized storage systems.

10.4 User Experience and Accessibility

7.4.1 Intuitive Interfaces

As steganography becomes more widely used, there is a need for tools with more intuitive and user-friendly interfaces. This includes simplifying the embedding and extraction processes, providing clear instructions, and offering support for a wider range of users.

Implication for Tool Development: Future steganography tools should focus on improving the user experience by designing intuitive interfaces and providing comprehensive documentation and support.

7.4.2 Cross-Platform Compatibility

Users expect tools to work seamlessly across different devices and operating systems. Developing cross-platform steganography tools is essential for reaching a wider audience.

Implication for Tool Development: Future tools should be designed with cross-platform compatibility in mind, using technologies that can be easily deployed on various operating systems and devices.

10.5 Conclusion

The future of steganography is marked by exciting developments and challenges. Advancements in deep learning, cloud computing, and security technologies are creating new opportunities for developing more powerful and sophisticated steganographic techniques. At the same time, the ongoing evolution of steganalysis and digital forensics requires constant vigilance and adaptation. By staying abreast of these trends and incorporating them into future tool development, we can ensure that steganography remains a valuable tool for secure communication and data protection in the years to come.